International Journal of Computer Networking, Wireless and Mobile Communications (IJCNWMC)
ISSN(P): 2250-1568; ISSN(E): 2278-9448
Vol. 4, Issue 1, Feb 2014, 59-70

TRANS STELLAR • Journal Publications • Research Consultancy

CONGESTION CONTROL TECHNIQUES AND PRINCIPLES IN COMPUTER NETWORKS: A SURVEY

ABHAY KUMAR[1], SMRITI AGRAWAL[2], NIRAJ UPADHYAYA[3] & A GOVARDHAN[4]

[1,2,3]J.B.Institute of Engineering & Technology, Hyderabad, Andhra Pradesh, India

[4]Jawaharlal Nehru Technological University, Hyderabad, Andhra Pradesh, India

ABSTRACT

Congestion control is one of the most important concerns in computer networks. There is a chance of inefficient usages of resources, possibly leading to network collapse if we do not use the proper congestion control algorithm. Therefore congestion control is an effort to readjust the performance of a network to fluctuations in the traffic load without adversely affecting user's perceived service quality. Computer networks are presently dominated by pure end-to-end feedback or implicit feedback congestion control. This approach has worked admirably but the networks are evolving and the traffic mix and amount has increased tremendously with the advent of multimedia applications in the Internet. Hence explicit help from within the network like switches or routers is crucial especially during congestion time. Active queue management (AQM) is a method which hand over congestion feedback from routers to end users. Explicit congestion notification (ECN) is an alternate approach which is more efficient because it does not have scalability problem. But the use of wireless devices in computer network have given interest to look for some methods related with explicit rate feedback. Explicit rate feedback congestion control can be made to scale if correct rate-calculation techniques is defined and used.

KEYWORDS: Network Protocols, TCP, Congestion Control, Slow-Start, AIMD

INTRODUCTION

Congestion occurs when resource demands exceeds the capacity available [1]. A network is defined as congested if the quality of service seen by user decreases, when load in the network increases. As an example of congestion, downloading a file today may take twice the time it took a day before. Congestion has become a normal problem for most of the Internet users hence when it takes more time to download; we say the network is congested. Congestion control is applicable to all packet switched networks including Internet. The Internet that we used is based on bursty traffic i.e. at some point of time there is less load than the capacity it can handle, at some point of time there are more load than the capacity, hence routers in the Internet, place extra packets in a buffer called as FIFO (First In First Out) queue. When queue is full, packets are dropped. Packets in a queue increases delay hence queues should be kept as small as possible. The network can be congested when queues grow, causing more delay and in worst case packet loss.

BACKGROUND

The goal of congestion control method is to use network as efficiently as possible i.e. obtain the maximum throughput while preserving a low loss rate and negligible delay. Congestion must be averted as it leads to growth of queue and queue growth increases delay and loss. The heterogeneity of link speed along with source to\destination path which travels several ISP (Internet Service Providers) boundaries can also be a cause of congestion.

Closed-Loop versus Open-Loop Control

Systems that use feedback are known as closed-loop congestion control whereas a system which does not use feedback is known as open-loop control. In computer networks, applying open-loop control will require a prior knowledge about the network. A network that is based on open-loop congestion control may use resource reservation that is a new flow will enter only if it is allowed by the admission control entity. In multi-service computer network like the Internet, where a diverse range of different applications is supported, prior knowledge about the network is difficult to be known in advance hence we need to add feedback to the control and therefore use of closed-loop control is desired.

Implicit Feedback Closed-Loop Control

The word implicit feedback means that measurement is taken at receiver and may be used to find what is happening within the network. A general purpose congestion control method should normally have sender adjusts its rate whenever feedback from the receiver arrives. When a packet travels from source host to destination host there may be three possibilities for a packet:

- Packet can be delayed
- Packet can be dropped
- Packet can be changed

Packet can be delayed due to various reasons: distance, queuing, processing at nodes etc. Similarly a packet can be dropped due to various reasons: a queue is exceeded, a user is not admitted, equipment failure etc. But a packet may be changed whenever there is a change in header or its content (payload). We can see that the word "queue" is mentioned whenever packet is delayed or dropped. Therefore, packet delay and packet drop are the two factors that can be used to indicate congestion

Source Rate Control Rule

Packet loss is the implicit feedback and packet loss is interpreted as a sign of congestion in wired networks. In response to packet-loss sender needs to follow a rule that is based on "decrease the rate in the event of packet loss and increase the rate otherwise". Now let us see how to increase or decrease this rate. Van Jacobson (1988) suggested that the control law should be AIMD (Additive Increase Multiplicative Decrease) [27] [28]. Let the rate of a sender at time 't' is denoted by x (t), y (t) represents the binary feedback with values 0 meaning "no congestion" and 1 meaning "congestion" and assuming linear control, the rate update function of the sender can be expressed as:

$$x(t+1) = a_i + b_i x(t) \quad \text{if } y(t) = 0$$

$$= a_d + b_d x(t) \quad \text{if } y(t) = 1$$

Where a_i, b_i, a_d and b_d are constants. The linear control has both additive (a) and multiplicative component (b). The postfix $_i$ and $_d$ represents increase and decrease component respectively. If we allow the influence of only one component at a time, then four different possibilities can arise:

(i) $a_i = 0$; $a_d = 0$; $b_i > 1$; $0 < b_d < 1$

Multiplicative increase when no congestion, multiplicative decrease when Congestion

i.e, Multiplicative Increase, Multiplicative Decrease (MIMD)

(ii) $a_i > 0$; $a_d < 0$; $b_i = 1$; $b_d = 1$

Additive increase when no congestion, additive decrease when congestion

i.e, Additive Increase, Additive Decrease (AIAD)

(iii) $a_i > 0$; $a_d = 0$; $b_i = 1$; $0 < b_d < 1$

Additive increase when no congestion, multiplicative decrease when congestion

i.e, Additive Increase, Multiplicative Decrease (AIMD)

(iv) $a_i = 0$; $a_d < 0$; $b_i > 1$; $b_d = 1$

Multiplicative increase when no congestion, additive decrease when congestion

i.e, Multiplicative Increase, Additive Decrease (MIAD)

In (Chin & Jain 1989), an algebraic proof shows that linear decrease policy should be multiplicative, and the linear increase policy should always have an additive component. Hence out of the four different possibilities, AIMD [28] [31] approaches perfectly towards efficiency and fairness. The source rate control rule that is implemented in the Internet basically is an AIMD variant. Two factors are important for AIMD to work:

- Window-based control
- Knowledge of the RTT (Round Trip Time)

Window-Based versus Rate-Based Control

There are two basic methods to control (reduce) the rate of a sender: rate based and window based. Both methods have advantages and disadvantages.

- In rate-based control, sender is aware of a specific data rate (bits/second), and the receiver or a router informs the sender of a new rate that sender must not exceed. Rate-based control is simpler, and is more convenient for streaming multimedia applications as it does not pause if no feedback arrives.
- In window-based control, a sender keeps track of a window i.e., a certain number of packets/bytes that a sender can send before new acknowledgement arrives. For example, if the window size is 8 and no new acknowledgement comes, the sender is allowed to send only 8 packets and then it must stop unless new acknowledgement comes.

Hence window-based control shows a certain type of wait-and-proceed behavior. Since the sender's behavior is strictly guided by the receipt or non-receipt of incoming feedback, the window-based control algorithm is used as self-clocking. The window-based control is better match for conservation of packets principle i.e., a new packet is sent into the network only when an old packet leaves the network. Hence from a network packet point of view window-based control is better suited for congestion control, since sender will stop sending if there is congestion within network as feedback will not arrive. But the main drawback of window-based control algorithm is that it may create traffic bursts, and to solve this sender or receiver should deliberately delay its reaction known as pacing.

RTT ESTIMATION

If reliable communication is required, a sender needs to retransmit the packets when they are lost due to congestion in the network. The general method to implement this is to number packets consecutively and the receiver

acknowledges each of them. But whenever an ACK (Acknowledgement) is not received within the expected time, the sender assumes that the packet is lost and retransmits it. This method is called ARQ (Automatic Repeat Request). But ARQ require a timer that is initialized to RTO (Retransmit Time Out) value when a packet is sent. Finding the correct RTO value is important for congestion control because packet loss is interpreted as a sign of congestion. Since the time from sending a packet to receiving the corresponding ACK is an RTT, the ideal value of RTO should be a function of an RTT, a most recent RTT measurement. Also RTT depends on things that happen within the network (delay in queues, path changes etc), hence relying solely on most recent measurement is too simple approach. Hence a prediction must be made using the history of RTT samples. As a common rule of thumb, RTT prediction should be conservative i.e. overestimating the RTT causes less harm than underestimating it.

VARIENT OF TCP

With the rapid growth and use of computer networks in our daily lives, the load within the network is increasing more than the capacity that it can handle. Hence implicit feedback discussed earlier where sender adjusts its sending rate on account of the feedback from the receiver may not be suitable for the following reasons:

- Relying only on packet loss means the sources always need to increase their rates until queues are full - it is a form of congestion control that first causes congestion and then reacts.
- Relying on implicit feedback means making assumptions about the network - interpreting packet loss as a sign of congestion works well if it is wired network but increasing use of wireless devices in the network poses a new challenge and the assumption that congestion occurs only due to packet loss will not work because bit errors are more frequent in wireless environment.

Variants of TCP, which are of interest, include those implemented already, yet to be implemented and employing conventional slow-start algorithm.

TCP Tahoe

The TCP Tahoe was released in 1988 by V. Jacobson in [1] being the first implementation of Transmission Control Protocol (TCP) to employ congestion control mechanism [30] [34]. Tahoe contains the AIMD (additive increase, multiplication decrease) being its control mechanism. Tahoe achieved congestion control through adjusting its windows size additively to increase and multiplicatively to decrease. AIMD at the initial stage increases windows size exponentially but, after a certain threshold, it switches to linear window size increase i.e., by one packet per RTT before congestion occurs (additive increase). At this point, Tahoe switches to the congestion avoidance state. If the ACK for a packet is not received before a time out, the threshold is reduced by half and the congestion window is reduced to one packet (multiplicative decrease). In summary, TCP Tahoe controls congestion as follows:

- When congestion window is below the threshold, the congestion window grows exponentially (slow-start state)
- When the congestion window is above the threshold the congestion window grows linearly (additive increase) i.e., congestion avoidance
- Whenever there is a timeout, the threshold is set to one half of the current congestion window and the congestion window is set to one while the packet is retransmitted (multiplicative decrease)
- Algorithms implemented are slow-start [34] and congestion avoidance.

TCP Reno

Proposal to modify Tahoe was given in [8] [34]. Like its predecessor, Reno sets its congestion window to one packet upon a time out (RTO). However, Reno extends its algorithm to include the fast retransmit mechanism. The fast retransmit involves the re-transmission of a dropped packet if three duplicate ACKs for a packet are received before the RTO. Reno also introduces the fast recovery mechanisms which prevent transmission to reenter the slow-start state after a fast retransmit. Instead the window size is halved and the threshold is adjusted accordingly and TCP remain in congestion avoidance until a timeout occurs. This is discussed in detail in [9]. TCP Reno became the standard TCP algorithm implemented in most computers. TCP Reno implements the algorithms namely slow-start, congestion avoidance (AIMD), fast retransmission and fast recovery procedure.

TCP NewReno

The NewReno TCP includes a change to the Reno algorithm at the sender end with a view to eliminate Reno's wait for a retransmit time-out whenever multiple packets are lost from a window [10]. This change modifies the sender's behavior during fast recovery. When this happens, NewReno does not exit from the fast recovery state as in the case of Reno, but waits for the receipt of all the outstanding ACKs for that window. The followings are the summary of NewReno fast recovery actions:

- It finds the maximum packets outstanding while entering fast recovery
- When a new ACK is received and it acknowledges all the outstanding packets, then fast recovery is exited and cwnd (congestion window) is set to half the value of ssthresh (slow-start threshold), then it transits to the congestion avoidance state. But, if a partial ACK is received, then, it assumes the next packet in the link is lost and tries to retransmit
- It exits fast recovery when all data in the window is acknowledged [11].

SACK (TCP with Selective Acknowledgements)

One challenge with the NewReno algorithm is its inability to detect other lost packets until the ACK for the first retransmitted packet was received. This implies that NewReno suffers from the fact that the detection of each packet loss takes one RTT. Hence Selective Acknowledgment (SACK) was proposed by [12]. SACK is an extension of TCP Reno and TCP NewReno [29]. It intends to solve two problems of TCP Reno and NewReno i.e., detection of multiple packet loss and retransmission of more than one lost packet per RTT. SACK retains the slow-start and fast retransmission of Reno. It also has the coarse grained time out of Tahoe. SACK algorithms specify that instead of cumulative acknowledgement of packets as contained in TCP Tahoe, Reno and NewReno, packets should be acknowledged selectively. This requires each ACK to contain an entry for which packet that is being acknowledged. This enables the sender to figure out which packets have been acknowledged and which ones are still outstanding. SACK specifies that whenever a sender enters into fast recovery state, a variable "pipe" be initiated and used to estimate the number of packets that are missing along the path. It then sets the size of cwnd to half its current size as usual. Each time an ACK is received, the size of the pipe is decreased by one and when a packet is transmitted or retransmitted, the pipe is increased by one. Whenever the size of the pipe becomes smaller then cwnd, it checks which packets are yet to be acknowledged and retransmits immediately. If there are no outstanding ACK, it sends a new packet. Thus the sender only sends new or retransmitted packet if the pipe is less the cwnd. This way SACK can send more than one lost packet in a single RTT. Use of pipe variable separates the decision of when to send a packet from which packet to send. One major disadvantage of the SACK algorithm is in its relative

difficulty in implementation of selective acknowledgement.

TCP Vegas

The TCP congestion control schemes that have been described so far use packet loss based approach to measure congestion. There is a class of congestion control algorithms that change its congestion window size based on end-to-end delay. This approach originated from [13] and is presented by [14] as TCP Vegas. The followings are the differences between TCP Vegas and TCP Reno:

- In the slow-start state, congestion control was incorporated by a deliberate delay in congestion window growth.
- It updates its congestion window based on end-to-end delay instead of using packet-loss as the window update parameter.

TCP Vegas extended Reno re-transmission strategy. It keeps track of when each packet was sent and calculates an estimate of RTT for each transmission. This is done by monitoring how long it took each ACK to get back to the sender. Whenever a duplicate ACK is received, it performs the following check:

If (current RTT > RTT estimate)

If this is true, it retransmits the packet without waiting for 3 duplicate ACK or a time out as in Reno [14]. Hence, Vegas solves the problem of not detecting lost packets when the window is very small i.e., less than three and could not receive enough duplicate ACKs. TCP Vegas congestion avoidance behavior is different from other TCP implementations. It determines congestion states using the sending rate. If there is a decrease in calculated rate of transmission as a result of large queue in the link, it reduces its window. When the sending rate increases, the window size also increases.

Summary of Various Congestion Control Protocols

Henceforth, TCP Tahoe, Reno and NewReno are referred to as NewReno. This type of congestion control algorithm is reactive in nature and is classified as loss based congestion control algorithm. This is the transport protocol of choice and it is implemented in over 90% of Internets traffic today. It became officially recognized in 2004 [20] [34]. However, some of congestion control algorithm are proactive in nature and are classified as delay based congestion control algorithm like TCP Vegas. To implement delay based congestion control, it is necessary to measure propagation delay. Propagation delay is the time it takes a packet to travel from the sender to its destination. The propagation delay is usually set to the smallest observed RTT. There are several observed problems with the estimation of the queuing delay. Estimating queuing delay is challenging if the RTT contains more elements then fixed propagation delay, e.g. retransmission delay in wireless link, a high loaded Internet link etc. There are very few research work done on delay based congestion control in wireless mobile network with the exception of [15] which proposed delay based congestion control scheme for commercial CDMA (Code Division Multiple Access).

Table 1: Variants of TCP Congestion Control Implementation

Protocol	Type	Purpose
TCP New-Reno [20]	Loss based	The standard TCP protocol
STCP [17]	Loss based	Higher throughput with high capacity and large delay
HSTCP [16]	Loss based	Higher throughput with high capacity and large delay
BIC – TCP [21]	Loss based	Higher throughput with high capacity and large delay
CUBIC [22]	Loss based	Higher throughput with high capacity and large delay
TCP Vegas [14]	Delay based	Higher through puts and reduced loss rate
Fast TCP [23]	Delay based	Higher throughput with high capacity and large delay
TCP Africa [24]	Loss delay based	Higher throughput with high capacity and large delay

Table 1: Contd.,

Compound TCP [18]	Loss delay based	Higher throughput with high capacity and large delay
TCP Illinois [19]	Loss delay based	Higher throughput with high capacity and large delay
West wood + [25]	Bandwidth estimation	Higher throughput over wireless networks. High capacity & large delay networks.
XCP [26]	Extra signaling	Higher throughput over wireless networks. Also for high capacity and large delays. Smaller queues, separate fairness control.

Other lines of researches are in the area of high-speed, large delay networks. Prominent among these are the High Speed TCP (HSTCP) proposed by [16] and scalable TCP (STCP) proposed by [17] which is experimental protocols that attempt to improve TCP performance under large bandwidth-delay product. They make TCP increments rule become more aggressive. Loss-delay based strategy was used by TCP Africa, Compound TCP [18] and TCP Illinois [19]. These protocols try to increase window size more aggressively than TCP NewReno as long as the network is not fully utilized and it switches to AIMD behavior of Reno when the network is near congestion.

Apart from the above mentioned congestion control protocols, there is Compound TCP [18] which is the version implemented in Microsoft window 7 operating system and is expected to grow in usage across the Internet. Table 1 shows the comparison between different congestion protocol algorithms.

OTHER TECHNIQUES FOR CONGESTION CONTROL

Apart from the above variant of TCP, there are other techniques as discussed below:

Active Queue Management (AQM)

Computer network is based on ‘store & forward’ switching technique (there is no fluid-flow model) where a fixed packets either reach a node or fail completely. Some flows may have the bad luck of transmitting packets at wrong time instances (based on RTT) i.e., when the queue is full. This problem is called as traffic phase effects. A FIFO queue is an example that can distort the output and the bandwidth may not be fairly divided among the flows. Traffic phase effect occurs when different flows see different performances. This problem can be solved by increasing the buffer or queue size in the router. But a queue is only meant to compensate for the sudden traffic burst. We have already seen that in general queues should be kept small because queuing delay should be as small as possible. The ideal maximum size of the queue should be “bandwidth * delay” product. Even if we choose the right queue length, the traffic phase effect will not vanish as control is still executed independent of the individual RTTs flows i.e., relying only on one such metric known as average RTT of all flows is not sufficient. Hence we need to introduce randomness in one of the controlling entities.

In the case of the Internet [32], the chosen entity was the router. Random Early Detection (RED) is a mechanism (Floyd and Jacobson 1993) which is widely used in routers and makes a decision to drop a packet on the basis of average queue length and a random function. RED is an example of active queue management (AQM) mechanism.

Explicit Congestion Notification (ECN)

Even with AQM schemes, help from within the network is not explicit i.e., end systems are not informed about what happen along the path between them. ECN is implemented in packet header using a single bit with ‘1’ indicating Congestion. ECN is used in large variety of Networks like Frame Relay, ATM and more recently in IP. The rule used by ECN method is that routers should set this bit instead of dropping packets when AQM methods decide to drop packet. Therefore end systems that see a set ECN bit should update (reduce) the rate as if the packet had been dropped. Apart from ECN that uses a single bit in the packet header, if we can increase the number of bits to two or more, then there can be

three basic methods to implement explicit feedback from within the network that are as follows:

- Choke packets
- Explicit rate feedback
- Backpressure

Choke Packets

In this method, a router sends a message "please reduce rate" to sources directly as soon as it encounters congestion. An example of this method known as Backward ECN (BECN) is used in frame relay.

Explicit Rate Feedback

In this method, the source node generates special packets or dedicated header fields in regular data packets, which need to be updated by the routers along the path. The example of this method is the ATM Available Bit Rate (ABR) service when source generates resource management (RM) cells which contain an 'Explicit rate' field. Each switch calculates the maximum allowed rate for the sender and reduces the field if it is more. Hence RM cell carries the smallest allowed rate of the path.

Backpressure

This method is also known as hop-by-hop congestion control, where each hop (router) along an end-to-end path sends feedback to the preceding router, which executes control based on this feedback.

COMPARISION OF THE TECHNIQUES

AQM schemes are used to manage queue length, reduce end-to-end latency, reduce packets dropping and avoid traffic phase effect within the network. But the design of such a scheme is a difficult task due to the range of possibilities to choose from: packets can be dropped from the front or the end of the queue, or they can be dropped from the middle of the queue but it is often inefficient because it leads to time consuming memory management operations. Also there are many different methods to monitor the queue size [35] and we need to use it in combination with some randomness to make a decision. Lastly the most important design goal of the random function chosen should be scalability. Hence in nutshell AQM is difficult to design properly because of the scalability requirements of computer network.ECN when used with AQM makes the system more efficient. ECN does not have scalability problem as additional effort for routers is almost negligible because a single bit in packet headers is used to implement it. Moreover ECN turns implicit feedback into explicit feedback with a signal available that conveys a clear message that cannot be misinterpreted. Also due to its explicit nature, it can be seen by any node that is traversed by packet that carries the signal.

If adding explicit feedback in the form of a single bit does not hurt the system, can we use two or more bits that would indicate different levels of congestion. The choke packet method also called as backward ECN is the fastest method to inform the sources to reduce the rate but cannot be implemented in large network due to following reasons: (i) when system is already congested, generating a packet is costly (ii) receiver is out of control loop hence RTT estimation can be a problem (iii) choke packet appear to limit the scalability of the network. Explicit rate feed back where source node generates dedicated header fields that must be updated by switches or routers along the path requires per-flow state and hence limit the scalability of the network. Lastly backpressure method that sends feedback directly to preceding switches or router imposes significant amount of work on switches or routers, may lead to deadlock and hence cannot be used in practice.

CONCLUSIONS

Computer networks especially Internet is presently dominated by pure end-to-end feedback or implicit feedback congestion control. The implicit feedback congestion control has advantages like it is lightweight and it is scalable but has some disadvantages like assumption of environment is made, it can't handle heterogeneous environment and is not designed to handle various operation modes like multicast congestion control etc.

But the success of explicit feedback congestion control especially ECN and increasing number of problems experienced with present methods as nature of network connections changes (like use of wireless in computer network) have given interest to look for some methods related with explicit rate feedback. Hence explicit rate feedback congestion control can be made to scale if correct rate calculation techniques is defined and used.

REFERENCES

1. V. Jacobson, "Congestion Avoidance and Control," *Proceedings of ACM Sigcomm*, Scanford, 26-30 August 1988, pp. 314-329.
2. R. Jain, "Congestion Control in Computer Networks: Issues and Trends," *IEEE Network Magazine*, May 1990, pp. 24-30.
3. W. R. Stevens, "TCP/IP Illustrated," *The Protocols*, Vol. 1, Addison-Wesley, 1997.
4. S. Floyd and F. Kevin, "Promoting the Use of End-to-End Congestion Control in the Internet," *IEEE/ACM Transactions on Networking*, May 3, 1999.
5. M. Welzl, "Network Congestion Control: Managing Internet Traffic," John Wiley & Sons Ltd., 2005.
6. B. A. Forouzan, "data Communications and Networking," Tata McGraw Hill, 2006.
7. B. A. Forouzan, "TCP/IP Protocol Suite," Tata McGraw Hill, 2006.
8. J. C. Hoe, "Improving the Start up Behaviour of a Congestion Control Scheme for TCP," *Proceedings of ACM Sigcomm*, 1996, pp. 270-280.
9. M. Allman, V. Paxson and W. R. Stevens, "TCP Congestion Control," RFC 2581, 1999. www.icir.org/mallman/papers/rfc2581.txt
10. K. Fall and S. Floyd, "Simulation Based Comparisons of Tahoe, Reno and SACK TCP," *ACM SIGCOMM Computer Communication Review*, Vol. 26, No. 3, July 1996.
11. Kolawole I. Oyeyinka, etl "TCP Window Based Congestion Control Slow Start Approach," *Communications and Network*, 2011, 3, 85-98, Published Online May 2011 (http://www.scirp.org/journal/cn)
12. M. Mathis, J. Mahdavi, S. Floyd and A. Romanow, "TCP Selective Acknowledgment Options," RFC 2018, Internet Engineering Task Force, October 1996. http://www.ietf.org/rfc/rfc2018.txt
13. R. Jaine, "A Delay-Based Approach for Congestion Avoidance in Interconnected Heterogeneous Computer Network," *ACM Computer Communication Review*, Vol. 19, No. 5, 1989. pp. 56-71.
14. L. Brakmo and L. Peterson, "TCP Vegas: End-to-End Congestion Avoidance on Global Internet," *IEEE Journal on Selected Areas in Communications*, Vol. 13, No. 8, 1995, pp. 1465-1480.
15. R. S. Jayaram and I. Rhee, "A Case of Delay-Based Flow Control in CDMA 2.5G Networks," *International*

Conference on Ubitiquitous Computing, Washington, 2003.

16. S. Floyd, "High Speed TCP for Large Congestion Windows," RFC 3649, 2003. www.ietf.org/rfc/rfc3649.txt
17. T. Kelly, "Scalable TCP Improving Performance in High Speed Wide Area Networks," *Computer Communications Review*, Vol. 32, No. 2, 2003, pp. 83-92.
18. K. Tang, J. M. Song, Q. Zhang and M. Sridharan, "A Compound TCP Approach for High-Speed and Long Distance Networks," *Proceedings of IEEE INFOCOM*, Barcelona, 23-29 April 2006.
19. S. Liu, T. Basar and R. Srikant, "TCP-Illinois: A Loss and Delay-Based Congestion Control Algorithm for High-Speed Networks," *Proceedings of First International Conference on Performance Evaluation Methodolgly Tools*, 2006.
20. S. Floyd and T. Henderson, "The New Reno Modification to TCP's Fast Recovery Algorithm," RFC 2582, Internet Engineering Task Force, April 1999. http.//www.ietf.org/rfc/rfc2528.txt
21. L. Xu, K. Harfoush and I. Rhee, "Binary Increase Con-gestion Control for Fast, Long Distance Networks," *Technology Report*, Computer Science Department, NC State University, Raleigh, 2003.
22. I. Rhee and L. Xu, "CUBIC: A New TCP—Friendly High Speed TCP Variant," *International Workshop on Protocols for Fast Long—Distance Networks*, Lyon, 3-4 February 2005.
23. D. X. Wei, C. Jin, S. H. Low, and S. Hedges, "Fast TCP: Motivation, Architecture, Algorithms, Performance," *IEEE- ACM Transaction on Networking*, Vol. 14, No. 6, 2006. pp. 1246-1259.
24. R. King, R. Baraniuk and R. Riedi, "TCP—Africa; an Adaptive and Fair Rapid Increase Rule for Scalable TCP," *Proceedings of IEEE INFOCOM*, Vol. 3, 2005, pp. 1838- 1848.
25. S. Mascolo, C. Casetti, M. Gerla, M. Y. Sanadidi and R. Wang, "TCP Westwood: Bandwidth Estimation for Enhanced Transport over Wireless Links," *Mobile Computing and Networking*, Rome, 16-21 July 2001, pp. 287-297.
26. S. H. Low and D. E. Lapsley, "Optimization Flow Control: Basic Algorithm and Convergence," *IEEE/ACM Transaction on Networking*, Vol. 7, No. 6, 1999, pp. 861-874.
27. B Subramani, Dr E Chandra, "A Survey on Congestion Control," *Global Journal of Computer Science and Technology*, Vol 9, No 5 (2010)
28. Hayder Natiq Jasem and al. "The New AIMD Congestion Control Algorithm," *Proceedings of WORLD ACADEMY OF SCIENCE, ENGINEERING AND TECHNOLOGY*, Vol 38, Feb 2009, ISSN: 2070-3740.
29. KaliprasadA.Mahapatro, MilindE.Rane, "A Novel Approach for Internet Congestion Control using an Extended State Observer," International Journal of Electronics and Communication Engineering & Technology (IJECET), ISSN 0976 –6464(Print), ISSN 0976 – 6472(Online) Volume 4, Issue 2, March – April (2013)
30. M. Crocker, Y. Chen, W. Hu, and W. Zhu, "A Survey of Reliable Transport Protocols," Intelligent Electronic Sysyems, Center for Advanced Vehicular Systems.
31. Jasem, Hayder Natiq and Zukarnain , Zuriati A. and Othman, Mohamed and Subramaniam , Shamala (2009), "Fairness Of the TCP-based new *AIMD congestion control algorith,". Journal of Theoretical and Applied Information Technology*, 5 (5). pp. 568-576. ISSN 1992-8645

32. Haider, Aun, “Improved congestion control for packet switched data networks and the Internet,” University of Canterbury, Electrical and Electronic Engineering, 2004.

33. D. Papadimitriou, M. Welzl, M. Scharf, B. Briscoe, “Open Research Issues in Internet Congestion Control,” RFC 6077, Feb 2011, ISSN: 2070-1721

34. K. Oyeyinka, A. Oluwatope, A. Akinwale, O. Folorunso, G. Aderounmu and O. Abiona, "TCP Window Based Congestion Control -Slow-Start Approach," *Communications and Network*, Vol. 3 No. 2, 2011, pp. 85-98. doi: 10.4236/cn.2011.32011, Scientific Research Open Access.

35. Ivan Marsic, “Computer Networks Performance and Quality of Service,” Department of Electrical and Computer Engineering, Rutgers University.